Version: **2023.01**

Last update: 2023/12/17 05:49

DOE603 - Managing and Storing Docker Images

Contents

LAB #1 - Re-creating an official docker image

1.1 - Using a Dockerfile

Although images are compiled by Docker Hub, it is possible to compile an “official” image from a Dockerfile:

```
root@debian11:~# mkdir mongodb
root@debian11:~# cd mongodb/
root@debian11:~/mongodb# touch Dockerfile docker-entrypoint.sh
```

The Docker file contains the instructions needed to build the image:

```
root@debian11:~/mongodb# vi Dockerfile
root@debian11:~/mongodb# cat Dockerfile
FROM ubuntu:bionic

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever
dependencies get added
RUN groupadd -r mongodb && useradd -r -g mongodb mongodb

RUN set -eux; \
        apt-get update; \
        apt-get install -y --no-install-recommends
                ca-certificates
                jq
                numactl
        ; \
        if ! command -v ps > /dev/null; then \
                apt-get install -y --no-install-recommends procps; \
        fi; \
        rm -rf /var/lib/apt/lists/*
```

```
# grab gosu for easy step-down from root (https://github.com/tianon/gosu/releases)
ENV GOSU_VERSION 1.11
# grab "js-yaml" for parsing mongod's YAML config files (https://github.com/nodeca/js-yaml/releases)
ENV JSYAML_VERSION 3.13.0

RUN set -ex; \
        \
        apt-get update; \
        apt-get install -y --no-install-recommends
                wget
        ; \
        if ! command -v gpg > /dev/null; then \
                apt-get install -y --no-install-recommends gnupg dirmngr; \
        fi; \
        rm -rf /var/lib/apt/lists/*; \
        \
        dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
        wget -O /usr/local/bin/gosu
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch"; \\
        wget -O /usr/local/bin/gosu.asc
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
        export GNUPGHOME="$(mktemp -d)"; \
        gpg --batch --keyserver pgp.mit.edu --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \
        # gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
        command -v gpgconf && gpgconf --kill all || :; \
        rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc; \
        chmod +x /usr/local/bin/gosu; \
        gosu --version; \
        gosu nobody true; \
        \
        wget -O /js-yaml.js "https://github.com/nodeca/js-yaml/raw/${JSYAML_VERSION}/dist/js-yaml.js"; \
# TODO some sort of download verification here
        \
        apt-get purge -y --auto-remove wget
```

```
RUN mkdir /docker-entrypoint-initdb.d

ENV GPG_KEYS E162F504A20CDF15827F718D4B7C549A058F8B6B
RUN set -ex; \
        export GNUPGHOME="$(mktemp -d)"; \
        for key in $GPG_KEYS; do \
                gpg --batch --keyserver pgp.mit.edu --recv-keys "$key"; \
        done; \
        gpg --batch --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg; \
        command -v gpgconf && gpgconf --kill all || :; \
        rm -r "$GNUPGHOME"; \
        apt-key list

# Allow build-time overrides (eg. to build image with MongoDB Enterprise version)
# Options for MONGO_PACKAGE: mongodb-org OR mongodb-enterprise
# Options for MONGO_REPO: repo.mongodb.org OR repo.mongodb.com
# Example: docker build --build-arg MONGO_PACKAGE=mongodb-enterprise --build-arg MONGO_REPO=repo.mongodb.com .
ARG MONGO_PACKAGE=mongodb-org-unstable
ARG MONGO_REPO=repo.mongodb.org
ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}

ENV MONGO_MAJOR 4.1
ENV MONGO_VERSION 4.1.9
# bashbrew-architectures:amd64 arm64v8 s390x
RUN echo "deb http://$MONGO_REPO/apt/ubuntu bionic/${MONGO_PACKAGE%-unstable}/$MONGO_MAJOR multiverse" | tee
"/etc/apt/sources.list.d/${MONGO_PACKAGE%-unstable}.list"

RUN set -x
        && apt-get update \
        && apt-get install -y \
                ${MONGO_PACKAGE}=$MONGO_VERSION \
                ${MONGO_PACKAGE}-server=$MONGO_VERSION \
                ${MONGO_PACKAGE}-shell=$MONGO_VERSION \y
                ${MONGO_PACKAGE}-mongos=$MONGO_VERSION \
```

```
                ${MONGO_PACKAGE}-tools=$MONGO_VERSION \
        && rm -rf /var/lib/apt/lists/* \
        && rm -rf /var/lib/mongodb \
        && mv /etc/mongod.conf /etc/mongod.conf.orig

RUN mkdir -p /data/db /data/configdb \
        && chown -R mongodb:mongodb /data/db /data/configdb
VOLUME /data/db /data/configdb

COPY docker-entrypoint.sh /usr/local/bin/
ENTRYPOINT ["docker-entrypoint.sh"]

EXHIBIT 27017
CMD ["mongod"]
```

The docker-entrypoint.sh file is used to launch the mongodb server in the container:

```
root@debian11:~/mongodb# vi docker-entrypoint.sh
root@debian11:~/mongodb# cat docker-entrypoint.sh
#!/bin/bash
set -Eeuo pipefail

if [ "${1:0:1}" = '-' ]; then
        set -- mongod "$@"
fi

originalArgOne="$1"

# allow the container to be started with `--user`
# all mongo* commands should be dropped to the correct user
if [[ "$originalArgOne" == mongo* ]] && [ "$(id -u)" = '0' ]; then
        if [ "$originalArgOne" = 'mongod' ]; then
                find /data/configdb /data/db ! -user mongodb -exec chown mongodb '{}' +
        fi
```

```
        # make sure we can write to stdout and stderr as "mongodb"
        # (for our "initdb" code later; see "--logpath" below)
        chown --dereference mongodb "/proc/$$/fd/1" "/proc/$$/fd/2" || :
        # ignore errors thanks to https://github.com/docker-library/mongo/issues/149

        exec gosu mongodb "$BASH_SOURCE" "$@"
fi

# you should use numactl to start your mongod instances, including the config servers, mongos instances, and any
clients.
# https://docs.mongodb.com/manual/administration/production-notes/#configuring-numa-on-linux
if [[ "$originalArgOne" == mongo* ]]; then
        numa='numactl --interleave=all'
        if $numa true &> /dev/null; then
                set -- $numa "$@"
        fi
fi

# usage: file_env VAR [DEFAULT]
# ie: file_env 'XYZ_DB_PASSWORD' 'example'
# (will allow for "$XYZ_DB_PASSWORD_FILE" to fill in the value of
# "$XYZ_DB_PASSWORD" from a file, especially for Docker's secrets feature)
file_env() {
        local var="$1"
        local fileVar="${var}_FILE"
        local def="${2:-}"
        if [ "${!var:-}" ] && [ "${!fileVar:-}" ]; then
                echo >&2 "error: both $var and $fileVar are set (but are exclusive)"
                exit 1
        fi
        local val="$def"
        if [ "${!var:-}" ]; then
                val="${!var}"
        elif [ "${!fileVar:-}" ]; then
```

```
                val="$(< "${!fileVar}")"
        fi
        export "$var"="$val"
        unset "$fileVar"
}

# see https://github.com/docker-library/mongo/issues/147 (mongod is picky about duplicated arguments)
_mongod_hack_have_arg() {
        local checkArg="$1"; shift
        local arg
        for arg; do
                case "$arg" in
                        "$checkArg"|"$checkArg"=*)
                                return 0
                                ;;
                esac
        done
        return 1
}
# _mongod_hack_get_arg_val '--some-arg' "$@"
_mongod_hack_get_arg_val() {
        local checkArg="$1"; shift
        while [ "$#" -gt 0 ]; do
                local arg="$1"; shift
                case "$arg" in
                        "$checkArg")
                                echo "$1"
                                return 0
                                ;;
                        "$checkArg"=*)
                                echo "${arg#$checkArg=}"
                                return 0
                                ;;
                esac
```

```
        done
        return 1
}
declare -a mongodHackedArgs
# _mongod_hack_ensure_arg '--some-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_arg() {
        local ensureArg="$1"; shift
        mongodHackedArgs=( "$@" )
        if ! _mongod_hack_have_arg "$ensureArg" "$@"; then
                mongodHackedArgs+=( "$ensureArg" )
        fi
}
# _mongod_hack_ensure_no_arg '--some-unwanted-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_no_arg() {
        local ensureNoArg="$1"; shift
        mongodHackedArgs=()
        while [ "$#" -gt 0 ]; do
                local arg="$1"; shift
                if [ "$arg" = "$ensureNoArg" ]; then
                        continue
                fi
                mongodHackedArgs+=( "$arg" )
        done
}
# _mongod_hack_ensure_no_arg '--some-unwanted-arg' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_no_arg_val() {
        local ensureNoArg="$1"; shift
        mongodHackedArgs=()
        while [ "$#" -gt 0 ]; do
                local arg="$1"; shift
                case "$arg" in
```

```
                        "$ensureNoArg")
                                shift # also skip the value
                                continue
                                ;;
                        "$ensureNoArg"=*)
                                # value is already included
                                continue
                                ;;
                esac
                mongodHackedArgs+=( "$arg" )
        done
}
# _mongod_hack_ensure_arg_val '--some-arg' 'some-val' "$@"
# set -- "${mongodHackedArgs[@]}"
_mongod_hack_ensure_arg_val() {
        local ensureArg="$1"; shift
        local ensureVal="$1"; shift
        _mongod_hack_ensure_no_arg_val "$ensureArg" "$@"
        mongodHackedArgs+=( "$ensureArg" "$ensureVal" )
}

# _js_escape 'some "string" value'
_js_escape() {
        jq --null-input --arg 'str' "$1" "$str'
}

jsonConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-config.json"
tempConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-temp-config.json"
_parse_config() {
        if [ -s "$tempConfigFile" ]; then
                return 0
        fi

        local configPath
```

```
        if configPath="$(_mongod_hack_get_arg_val --config "$@")"; then
                # if --config is specified, parse it into a JSON file so we can remove a few problematic keys
(especially SSL-related keys)
                # see https://docs.mongodb.com/manual/reference/configuration-options/
                mongo --norc --nodb --quiet --eval "load('/js-yaml.js'); printjson(jsyaml.load(cat($(_js_escape
"$configPath"))))"" > "$jsonConfigFile"
                jq 'del(.systemLog, .processManagement, .net, .security)' "$jsonConfigFile" > "$tempConfigFile"
                return 0
        fi

        return 1
}
dbPath=
_dbPath() {
        if [ -n "$dbPath" ]; then
                echo "$dbPath"
                return
        fi

        if ! dbPath="$(_mongod_hack_get_arg_val --dbpath "$@")"; then
                if _parse_config "$@"; then
                        dbPath="$(jq -r '.storage.dbPath // empty' "$jsonConfigFile")"
                fi
        fi

        if [ -z "$dbPath" ]; then
                if _mongod_hack_have_arg --configsvr "$@" || {
                        _parse_config "$@" \
                        && clusterRole="$(jq -r '.sharding.clusterRole // empty' "$jsonConfigFile")" \
                        && [ "$clusterRole" = 'configsvr']
                }; then
                        # if running as config server, then the default dbpath is /data/configdb
                        # https://docs.mongodb.com/manual/reference/program/mongod/#cmdoption-mongod-configsvr
                        dbPath=/data/configdb
```

```
                fi
        fi

        : "${dbPath:=/data/db}"

        echo "$dbPath"
}

if [ "$originalArgOne" = 'mongod' ]; then
        file_env 'MONGO_INITDB_ROOT_USERNAME'
        file_env 'MONGO_INITDB_ROOT_PASSWORD'
        # pre-check a few factors to see if it's even worth bothering with initdb
        shouldPerformInitdb=
        if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
                # if we have a username/password, let's set "--auth"
                _mongod_hack_ensure_arg '--auth' "$@"
                set -- "${mongodHackedArgs[@]}"
                shouldPerformInitdb='true'
        elif [ "$MONGO_INITDB_ROOT_USERNAME" ] || [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
                cat >&2 <<-'EOF'
                        error: missing 'MONGO_INITDB_ROOT_USERNAME' or 'MONGO_INITDB_ROOT_PASSWORD'
                               both must be specified for a user to be created
                EOF
                exit 1
        fi

        if [ -z "$shouldPerformInitdb" ]; then
                # if we've got any /docker-entrypoint-initdb.d/* files to parse later, we should initdb
                for f in /docker-entrypoint-initdb.d/*; do
                        case "$f" in
                                *.sh|*.js) # this should match the set of files we check for below
                                        shouldPerformInitdb="$f"
                                        break
                                        ;;
```

```
                        esac
                done
        fi

        # check for a few known paths (to determine whether we've already initialized and should thus skip our
initdb scripts)
        if [ -n "$shouldPerformInitdb" ]; then
                dbPath="$(_dbPath "$@")"
                for path in \
                        "$dbPath/WiredTiger" \
                        "$dbPath/journal" \
                        "$dbPath/local.0" \
                        "$dbPath/storage.bson" \
                ; do
                        if [ -e "$path" ]; then
                                shouldPerformInitdb=
                                break
                        fi
                done
        fi

        if [ -n "$shouldPerformInitdb" ]; then
                mongodHackedArgs=( "$@" )
                if _parse_config "$@"; then
                        _mongod_hack_ensure_arg_val --config "$tempConfigFile" "${mongodHackedArgs[@]}"
                fi
                _mongod_hack_ensure_arg_val --bind_ip 127.0.0.1 "${mongodHackedArgs[@]}"
                _mongod_hack_ensure_arg_val --port 27017 "${mongodHackedArgs[@]}"
                _mongod_hack_ensure_no_arg --bind_ip_all "${mongodHackedArgs[@]}"

                # remove "--auth" and "--replSet" for our initial startup (see
https://docs.mongodb.com/manual/tutorial/enable-authentication/#start-mongodb-without-access-control)
                # https://github.com/docker-library/mongo/issues/211
                _mongod_hack_ensure_no_arg --auth "${mongodHackedArgs[@]}"
```

```
                if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
                        _mongod_hack_ensure_no_arg_val --replSet "${mongodHackedArgs[@]}"
                fi

                sslMode="$(_mongod_hack_have_arg '--sslPEMKeyFile' "$@" && echo 'allowSSL' || echo 'disabled')" #
"BadValue: need sslPEMKeyFile when SSL is enabled" vs "BadValue: need to enable SSL via the sslMode flag when
using SSL configuration parameters"
                _mongod_hack_ensure_arg_val --sslMode "$sslMode" "${mongodHackedArgs[@]}"

                if stat "/proc/$$/fd/1" > /dev/null && [ -w "/proc/$$/fd/1" ]; then
                        #
https://github.com/mongodb/mongo/blob/38c0eb538d0fd390c6cb9ce9ae9894153f6e8ef5/src/mongo/db/initialize_server_glo
bal_state.cpp#L237-L251
                        # https://github.com/docker-library/mongo/issues/164#issuecomment-293965668
                        _mongod_hack_ensure_arg_val --logpath "/proc/$$/fd/1" "${mongodHackedArgs[@]}"
                else
                        initdbLogPath="$(_dbPath "$@")/docker-initdb.log"
                        echo >&2 "warning: initdb logs cannot write to '/proc/$$/fd/1', so they are in
'$initdbLogPath' instead"
                        _mongod_hack_ensure_arg_val --logpath "$initdbLogPath" "${mongodHackedArgs[@]}"
                fi
                _mongod_hack_ensure_arg --logappend "${mongodHackedArgs[@]}"

                pidfile="${TMPDIR:-/tmp}/docker-entrypoint-temp-mongod.pid"
                rm -f "$pidfile"
                _mongod_hack_ensure_arg_val --pidfilepath "$pidfile" "${mongodHackedArgs[@]}"

                "${mongodHackedArgs[@]}" --fork

                mongo=( mongo --host 127.0.0.1 --port 27017 --quiet )

                # check to see that our "mongod" actually did start up (catches "--help", "--version", MongoDB
3.2 being silly, slow prealloc, etc)
                # https://jira.mongodb.org/browse/SERVER-16292
```

```
                tries=30
                while true; do
                        if ! { [ -s "$pidfile" ] && ps "$(< "$pidfile")" &> /dev/null; }; then
                                # bail ASAP if "mongod" isn't even running
                                echo >&2
                                echo >&2 "error: $originalArgOne does not appear to have stayed running --
perhaps it had an error?"
                                echo >&2
                                exit 1
                        fi
                        if "${mongo[@]}" 'admin' --eval 'quit(0)' &> /dev/null; then
                                # success!
                                break
                        fi
                        (( tries-- ))
                        if [ "$tries" -le 0 ]; then
                                echo >&2
                                echo >&2 "error: $originalArgOne does not appear to have accepted connections
quickly enough -- perhaps it had an error?"
                                echo >&2
                                exit 1
                        fi
                        sleep 1
                done

                if [ "$MONGO_INITDB_ROOT_USERNAME" ] && [ "$MONGO_INITDB_ROOT_PASSWORD" ]; then
                        rootAuthDatabase='admin'

                        "${mongo[@]}" "$rootAuthDatabase" <<-EOJS
                                db.createUser({
                                        user: $(_js_escape "$MONGO_INITDB_ROOT_USERNAME"),
                                        pwd: $(_js_escape "$MONGO_INITDB_ROOT_PASSWORD"),
                                        roles: [ { role: 'root', db: $(_js_escape "$rootAuthDatabase") } ]
                                })
```

```
                        EOJS
                fi

                export MONGO_INITDB_DATABASE="${MONGO_INITDB_DATABASE:-test}"

                echo
                for f in /docker-entrypoint-initdb.d/*; do
                        case "$f" in
                                *.sh) echo "$0: running $f"; . "$f" ;;
                                *.js) echo "$0: running $f"; "${mongo[@]}" "$MONGO_INITDB_DATABASE" "$f"; echo ;;
                                *) echo "$0: ignoring $f" ;;
                        esac
                        echo
                done

                "${mongodHackedArgs[@]}" --shutdown
                rm -f "$pidfile"

                echo
                echo 'MongoDB init process complete; ready for start up.'
                echo
        fi

        # MongoDB 3.6+ defaults to localhost-only binding
        if mongod --help 2>&1 | grep -q -- --bind_ip_all; then # TODO remove this conditional when 3.4 is no
longer supported
                haveBindIp=
                if _mongod_hack_have_arg --bind_ip "$@" || _mongod_hack_have_arg --bind_ip_all "$@"; then
                        haveBindIp=1
                elif _parse_config "$@" && jq --exit-status '.net.bindIp // .net.bindIpAll' "$jsonConfigFile" >
/dev/null; then
                        haveBindIp=1
                fi
                if [ -z "$haveBindIp" ]; then
```

```
                        # so if no "--bind_ip" is specified, let's add "--bind_ip_all"
                        set -- "$@" --bind_ip_all
                fi
        fi

        unset "${!MONGO_INITDB_@}"
fi

rm -f "$jsonConfigFile" "$tempConfigFile"

exec "$@"
```

Let's examine each command in the Dockerfile:

1.2 - FROM

```
FROM ubuntu:bionic
```

This line defines the image from which our image will be built. When the image is not built from another image, the value of **FROM** is **scratch**.

1.3 - RUN

```
...

RUN groupadd -r mongodb && useradd -r -g mongodb

RUN set -eux; \
    apt-get update; \
    apt-get install -y --no-install-recommends
        ca-certificates
        jq
```

```
        numactl
    ; \
    if ! command -v ps > /dev/null; then \
        apt-get install -y --no-install-recommends procps; \
    fi; \
    rm -rf /var/lib/apt/lists/*
...
RUN set -ex; \
    \
    apt-get update; \
    apt-get install -y --no-install-recommends
        wget
    ; \
    if ! command -v gpg > /dev/null; then \
        apt-get install -y --no-install-recommends gnupg dirmngr; \
    fi; \
    rm -rf /var/lib/apt/lists/*; \
    \
    dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
    wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch";
\\
    wget -O /usr/local/bin/gosu.asc
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
    export GNUPGHOME="$(mktemp -d)"; \
    gpg --batch --keyserver pgp.mit.edu --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \
    gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc; \
    chmod +x /usr/local/bin/gosu; \
    gosu --version; \
    gosu nobody true; \
    \
    wget -O /js-yaml.js "https://github.com/nodeca/js-yaml/raw/${JSYAML_VERSION}/dist/js-yaml.js"; \
# TODO some sort of download verification here
```

```
    \
    apt-get purge -y --auto-remove wget

RUN mkdir /docker-entrypoint-initdb.d
...

RUN set -ex; \
    export GNUPGHOME="$(mktemp -d)"; \
    for key in $GPG_KEYS; do \
        gpg --batch --keyserver pgp.mit.edu --recv-keys "$key"; \
    done; \
    gpg --batch --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg; \
    command -v gpgconf && gpgconf --kill all || :; \
    rm -r "$GNUPGHOME"; \
    apt-key list
...
RUN set -x \
    && apt-get update \
    && apt-get install -y \
        ${MONGO_PACKAGE}=$MONGO_VERSION \
        ${MONGO_PACKAGE}-server=$MONGO_VERSION \
        ${MONGO_PACKAGE}-shell=$MONGO_VERSION \y
        ${MONGO_PACKAGE}-mongos=$MONGO_VERSION \
        ${MONGO_PACKAGE}-tools=$MONGO_VERSION \
    && rm -rf /var/lib/apt/lists/* \
    && rm -rf /var/lib/mongodb \
    && mv /etc/mongod.conf /etc/mongod.conf.orig

RUN mkdir -p /data/db /data/configdb \
    && chown -R mongodb:mongodb /data/db /data/configdb
...
```

This command launches a process in the image construction. In the cases above, each string corresponds to the command passed to the **/bin/sh** shell.

There is another syntax for the RUN command called the exec format, namely:

```
RUN ["/bin/bash", "-c", "command"]
```

> **Important**: The RUN command is used to execute a command passed as an argument when compiling the image only. This command should therefore not be used to execute a command when launching the container. The command used to accomplish the latter is ENTRYPOINT.

1.4 - ENV

This command is used to set the value of an environment variable available in the Dockerfile suite:

```
...
ENV GOSU_VERSION 1.11
# grab "js-yaml" for parsing mongod's YAML config files (https://github.com/nodeca/js-yaml/releases)
ENV JSYAML_VERSION 3.13.0
...

ENV GPG_KEYS E162F504A20CDF15827F718D4B7C549A058F8B6B
...

ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}

ENV MONGO_MAJOR 4.1
ENV MONGO_VERSION 4.1.95
...
```

and in containers generated from the constructed image.

1.5 - VOLUME

```
...
VOLUME /data/db /data/configdb
...
```

This command exposes the directories passed as arguments so that they can be mapped to directories on the host machine or elsewhere, as we saw with the nginx example.

1.6 - COPY

```
...
COPY docker-entrypoint.sh /usr/local/bin/
...
```

This command retrieves the files in the context and copies them to the image.

Please note: all files in the context are included in the final image, even those that are unnecessary.

It is possible to exclude files present in the context by putting them in a file called **.dockerignore** placed in the context.

> **Important** - There is another command similar to COPY: ADD. ADD is a command that is no longer recommended except in specific cases. Note that when using the ADD command, if the source file is a TAR archive, its contents will be unarchived and copied to the destination whereas if the source file is referenced by a URL, the contents will be downloaded and then dropped into the destination.

1.7 - ENTRYPOINT

```
...
ENTRYPOINT ["docker-entrypoint.sh"]
...
```

This command stipulates the command that will be executed when the container is started.

There are two possible scenarios:

- ENTRYPOINT followed by a string - a shell is started to execute the string,
- ENTRYPOINT followed by a JSON table (as above) in ENTRYPOINT format [“command to execute”, “command parameters”].

In the **docker-entrypoint.sh** file:

```
...
originalArgOne="$1"

# allow the container to be started with `--user`
# all mongo* commands should be dropped to the correct user
if [[ "$originalArgOne" == mongo* ]] && [ "$(id -u)" = '0' ]; then
    if [ "$originalArgOne" = 'mongod' ]; then
        find /data/configdb /data/db ! -user mongodb -exec chown mongodb '{}' +
    fi

    # make sure we can write to stdout and stderr as "mongodb"
    # (for our "initdb" code later; see "--logpath" below)
    chown --dereference mongodb "/proc/$$/fd/1" "/proc/$$/fd/2" || :
    # ignore errors thanks to https://github.com/docker-library/mongo/issues/149

    exec gosu mongodb "$BASH_SOURCE" "$@"
fi

# you should use numactl to start your mongod instances, including the config servers, mongos instances, and any
```

```
clients.
# https://docs.mongodb.com/manual/administration/production-notes/#configuring-numa-on-linux
if [[ "$originalArgOne" == mongo* ]]; then
    numa='numactl --interleave=all'
    if $numa true &> /dev/null; then
        set -- $numa "$@"
    fi
fi
...
exec "$@"
```

if the value of the parameter passed to entrypoint.sh is **mongod**, the script assigns the user mongodb to the directories /data/configdb and /data/db then launches mongo under the user mongodb with reduced rights (gosu).

This file ends with “$@” which indicates that if no condition has been met, the command is executed with the value passed as an argument.

> **Important** - Note that an image is compiled inside a **context**. The **context** is the build directory. Lastly, note that there can be several ENTRYPOINTs in the Dockerfile but only the last one is taken into account.

1.8 - EXPOSE

```
...
EXHIBIT 27017
...
```

This command exposes a port outside the container.

1.9 - CMD

```
...
CMD ["mongod"]
...
```

This represents the value of the default parameter if no parameter is specified at the end of the docker run command.

1.10 - Other Commands

The Dockerfile can also contain the following commands:

- **WORKDIR**,
  - This command sets the working directory when compiling an image. It can appear several times in the Dockerfile, allowing the working directory to change,
- **LABEL**,
  - This command allows you to define key/value pairs to be included in the metadata describing the image when it is distributed, for example, the **version**, the **description** or a **readme**.

Compile the image:

```
root@debian11:~/mongodb# docker build .
[+] Building 56.9s (15/15) FINISHED
docker:default
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 3.55kB
0.0s
```

```
 => [internal] load metadata for docker.io/library/ubuntu:bionic
0.3s
 => [internal] load build context
0.0s
 => => transferring context: 42B
0.0s
 => [ 1/10] FROM
docker.io/library/ubuntu:bionic@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
0.0s
 => CACHED [ 2/10] RUN groupadd -r mongodb && useradd -r -g mongodb mongodb
0.0s
 => CACHED [ 3/10] RUN set -eux;  apt-get update;  apt-get install -y --no-install-recommends   ca-certificates
jq   numactl  ;  if ! command -v ps > /dev/null; then   ap  0.0s
 => [ 4/10] RUN set -ex;   apt-get update;  apt-get install -y --no-install-recommends   wget  ;  if ! command -v
gpg > /dev/null; then   apt-get install -y --no-install-r  20.6s
 => [ 5/10] RUN mkdir /docker-entrypoint-initdb.d
0.5s
 => [ 6/10] RUN set -ex;  export GNUPGHOME="$(mktemp -d)";  for key in E162F504A20CDF15827F718D4B7C549A058F8B6B;
do   gpg --batch --keyserver pgp.mit.edu --recv-keys "$key  10.4s
 => [ 7/10] RUN echo "deb http://$MONGO_REPO/apt/ubuntu bionic/${MONGO_PACKAGE%-unstable}/$MONGO_MAJOR
multiverse" | tee "/etc/apt/sources.list.d/${MONGO_PACKAGE%-unstable}  0.5s
 => [ 8/10] RUN set -x  && apt-get update  && apt-get install -y   mongodb-org-unstable=4.1.9   mongodb-org-
unstable-server=4.1.9   mongodb-org-unstable-shell=4.1.9   mong  21.1s
 => [ 9/10] RUN mkdir -p /data/db /data/configdb  && chown -R mongodb:mongodb /data/db /data/configdb
0.5s
 => [10/10] COPY docker-entrypoint.sh /usr/local/bin/
0.1s
 => exporting to image
2.6s
 => => exporting layers
2.6s
 => => writing image sha256:72fad0b7e0c2206f31a12b7d49f0812c0a594a51e17a8c0e36687f5f626bc735
0.0s
```

View the list of images:

```
root@debian11:~/mongodb# docker images
REPOSITORY           TAG       IMAGE ID       CREATED              SIZE
<none>               <none>    72fad0b7e0c2   About a minute ago   352MB
ittraining/mongodb   latest    fb3c6d5d186a   7 hours ago          1.11GB
ubuntu               latest    b6548eacb063   9 days ago           77.8MB
nginx                latest    a6bd71f48f68   2 weeks ago          187MB
hello-world          latest    9c7a54a9a43c   7 months ago         13.3kB
centos               latest    5d0da3dc9764   2 years ago          231MB
```

Note that the image has no REPOSITORY or TAG. So create a TAG:

```
root@debian11:~/mongodb# docker tag 72f i2tch/mongodb1

root@debian11:~/mongodb# docker images
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
i2tch/mongodb1       latest    72fad0b7e0c2   2 minutes ago   352MB
ittraining/mongodb   latest    fb3c6d5d186a   7 hours ago     1.11GB
ubuntu               latest    b6548eacb063   9 days ago      77.8MB
nginx                latest    a6bd71f48f68   2 weeks ago     187MB
hello-world          latest    9c7a54a9a43c   7 months ago    13.3kB
centos               latest    5d0da3dc9764   2 years ago     231MB
```

Boot a container from the i2tch/mongodb1 image:

```
root@debian11:~/mongodb# docker run -d --name mongo1 i2tch/mongodb1
3c578ea2a0428a07b60dac3b63d806351dffa2bb05224bcf7d12f1189766f38e
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime
create failed: runc create failed: unable to start container process: exec: "docker-entrypoint.sh": executable
file not found in $PATH: unknown.

root@debian11:~/mongodb# ls -l
total 16
```

```
-rw-r--r-- 1 root root 10971 Dec 10 16:57 docker-entrypoint.sh
-rw-r--r-- 1 root root  3514 Dec 10 17:09 Dockerfile
```

Important - Note that the docker-entrypoint.sh file is not executable!

Recompile the image:

```
root@debian11:~/mongodb# docker rm mongo1
mongo1

root@debian11:~/mongodb# chmod +x docker-entrypoint.sh

root@debian11:~/mongodb# docker build .
[+] Building 0.8s (15/15) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 3.55kB
0.0s
 => [internal] load .dockerignore
0.1s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/library/ubuntu:bionic
0.3s
 => [ 1/10] FROM
docker.io/library/ubuntu:bionic@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
0.0s
 => [internal] load build context
0.0s
 => => transferring context: 11.02kB
```

```
0.0s
 => CACHED [ 2/10] RUN groupadd -r mongodb && useradd -r -g mongodb mongodb
0.0s
 => CACHED [ 3/10] RUN set -eux;  apt-get update;  apt-get install -y --no-install-recommends   ca-certificates
jq   numactl  ;  if ! command -v ps > /dev/null; then   ap  0.0s
 => CACHED [ 4/10] RUN set -ex;   apt-get update;  apt-get install -y --no-install-recommends   wget  ;  if !
command -v gpg > /dev/null; then   apt-get install -y --no-ins  0.0s
 => CACHED [ 5/10] RUN mkdir /docker-entrypoint-initdb.d
0.0s
 => CACHED [ 6/10] RUN set -ex;  export GNUPGHOME="$(mktemp -d)";  for key in
E162F504A20CDF15827F718D4B7C549A058F8B6B; do   gpg --batch --keyserver pgp.mit.edu --recv-keys  0.0s
 => CACHED [ 7/10] RUN echo "deb http://$MONGO_REPO/apt/ubuntu bionic/${MONGO_PACKAGE%-unstable}/$MONGO_MAJOR
multiverse" | tee "/etc/apt/sources.list.d/${MONGO_PACKAGE%-un  0.0s
 => CACHED [ 8/10] RUN set -x  && apt-get update  && apt-get install -y   mongodb-org-unstable=4.1.9   mongodb-
org-unstable-server=4.1.9   mongodb-org-unstable-shell=4.1.9   0.0s
 => CACHED [ 9/10] RUN mkdir -p /data/db /data/configdb  && chown -R mongodb:mongodb /data/db /data/configdb
0.0s
 => [10/10] COPY docker-entrypoint.sh /usr/local/bin/
0.2s
 => exporting to image
0.1s
 => => exporting layers
0.1s
 => => writing image sha256:56e5b1fb4284e2474392238ee5f91a5d27d9a4a43fa15f655136ae0283d269c2
0.0s
```

Important - Note the **CACHED** lines here. However, it is possible not to use the cache by stipulating **–no-cache**. Note also the use of temporary containers per new step, with a commit to an image and deletion of the container. Finally, note that an image is compiled within a **context**. The **context** is the build directory. **Please note**: all files in the context are included in the final image, even unnecessary ones.

Check the list of images again and rename your last image:

```
root@debian11:~/mongodb# docker images
REPOSITORY           TAG       IMAGE ID       CREATED              SIZE
<none>               <none>    56e5b1fb4284   About a minute ago   352MB
i2tch/mongodb1       latest    72fad0b7e0c2   5 minutes ago        352MB
ittraining/mongodb   latest    fb3c6d5d186a   7 hours ago          1.11GB
ubuntu               latest    b6548eacb063   9 days ago           77.8MB
nginx                latest    a6bd71f48f68   2 weeks ago          187MB
hello-world          latest    9c7a54a9a43c   7 months ago         13.3kB
centos               latest    5d0da3dc9764   2 years ago          231MB

root@debian11:~/mongodb# docker tag 56e i2tch/mongodb2

root@debian11:~/mongodb# docker images
REPOSITORY           TAG       IMAGE ID       CREATED              SIZE
i2tch/mongodb2       latest    56e5b1fb4284   About a minute ago   352MB
i2tch/mongodb1       latest    72fad0b7e0c2   5 minutes ago        352MB
ittraining/mongodb   latest    fb3c6d5d186a   7 hours ago          1.11GB
ubuntu               latest    b6548eacb063   9 days ago           77.8MB
nginx                latest    a6bd71f48f68   2 weeks ago          187MB
hello-world          latest    9c7a54a9a43c   7 months ago         13.3kB
centos               latest    5d0da3dc9764   2 years ago          231MB
```

Launch a container from the last image:

```
root@debian11:~/mongodb# docker run -d --name mongo2 i2tch/mongodb2
880733c6bdc33a9a8fa6ae171e977cf745ea9a1b9cfc914992a2d0d3f8cd9d39
```

Use the **docker ps** command to see if the mongodb process is started:

```
root@debian11:~/mongodb# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
NAMES
```

```
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   15 seconds ago   Up 13 seconds   27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   7 hours ago      Up 7 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   7 hours ago      Up 7 hours      0.0.0.0:81->80/tcp,
:::81->80/tcp   quirky_moore
```

Connect to mongodb from your host machine:

```
root@debian11:~/mongodb# docker inspect mongo2 | grep IP
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.4",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.4",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,

root@debian11:~/mongodb# mongo --host 172.17.0.4
MongoDB shell version v4.0.28
connecting to: mongodb://172.17.0.4:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("057eacfe-5b02-4653-9b20-a2a2044cbe6a") }
MongoDB server version: 4.1.9
WARNING: shell and server versions do not match
Server has startup warnings:
2023-12-10T16:16:13.395+0000 I STORAGE [initandlisten]
```

```
2023-12-10T16:16:13.395+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly
recommended with the WiredTiger storage engine
2023-12-10T16:16:13.395+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten]
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten] ** NOTE: This is a development version (4.1.9) of MongoDB.
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten] ** Not recommended for production.
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten]
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is
unrestricted.
2023-12-10T16:16:14.255+0000 I CONTROL [initandlisten]
2023-12-10T16:16:14.256+0000 I CONTROL [initandlisten]
2023-12-10T16:16:14.256+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is
'always'.
2023-12-10T16:16:14.256+0000 I CONTROL [initandlisten] ** We suggest setting it to 'never'
2023-12-10T16:16:14.256+0000 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> exit
bye
root@debian11:~/mongodb#
```

LAB #2 - Creating a Dockerfile

====2.1 - Creating and testing the==== script

Create a directory named myDocker:

```
root@debian11:~/mongodb# mkdir ~/myDocker
root@debian11:~/mongodb# cd ~/myDocker
root@debian11:~/myDocker#
```

Create the myEntrypoint.sh file:

```
root@debian11:~/myDocker# vi myEntrypoint.sh

root@debian11:~/myDocker# cat myEntrypoint.sh
#!/bin/bash
if [ -z "$myVariable" ]; then
        echo "The variable myVariable must have a value"
        return 1
fi

while true;
do
        echo $1 \($(date +%H:%M:%S)\);
        sleep "$myVariable";
done
```

Test this script:

```
root@debian11:~/myDocker# myVariable=3 . ./myEntrypoint.sh Hello!
Hello! (18:01:54)
Hello! (18:01:57)
Hello! (18:02:00)
```

```
Hello! (18:02:03)
Hello! (18:02:06)
^C
root@debian11:~/myDocker#
```

Make this script executable:

```
root@debian11:~/myDocker# chmod u+x myEntrypoint.sh
```

Now create the **Dockerfile** file in the **~/myDocker** directory:

```
root@debian11:~/myDocker# vi Dockerfile

root@debian11:~/myDocker# cat Dockerfile
FROM centos:latest
MAINTAINER Team IT Training "infos@ittraining.team"
COPY myEntrypoint.sh /entrypoint.sh
ENV myVariable 3
ENTRYPOINT ["/entrypoint.sh"]
CMD ["mycommand"]
```

Now generate the image:

```
root@debian11:~/myDocker# docker build -t i2tch/mydocker .
[+] Building 0.8s (7/7) FINISHED
docker:default
 => [internal] load .dockerignore
0.2s
 => => transferring context: 2B
0.0s
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 211B
0.0s
```

```
 => [internal] load metadata for docker.io/library/centos:latest
0.0s
 => [internal] load build context
0.1s
 => => transferring context: 224B
0.0s
 => [1/2] FROM docker.io/library/centos:latest
0.1s
 => [2/2] COPY myEntrypoint.sh /entrypoint.sh
0.2s
 => exporting to image
0.1s
 => => exporting layers
0.1s
 => => writing image sha256:c5a41438d278439fac2cd65d53d87cabc5c771dd9b99be1913ce049024eba961
0.0s
 => => naming to docker.io/i2tch/mydocker
0.0s
```

Launch the container:

```
root@debian11:~/myDocker# docker run -it --name myDocker i2tch/mydocker
mycommand (17:05:57)
mycommand (17:06:00)
mycommand (17:06:03)
^Cmycommand (17:06:06)
mycommand (17:06:09)
mycommand (17:06:12)
^P^Q
root@debian11:~/myDocker#
```

Important - Note that **^C** has no effect. To detach from the container you

should use **^P^Q**.

Note that the container is still running:

```
root@debian11:~/myDocker# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
NAMES
97fe360bb1d6   i2tch/mydocker       "/entrypoint.sh myco…"   4 minutes ago    Up 4 minutes
myDocker
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   54 minutes ago   Up 54 minutes   27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   8 hours ago      Up 8 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   8 hours ago      Up 8 hours      0.0.0.0:81->80/tcp,
:::81->80/tcp   quirky_moore

root@debian11:~/myDocker# docker logs myDocker | tail
mycommand (17:10:30)
mycommand (17:10:33)
mycommand (17:10:36)
mycommand (17:10:39)
mycommand (17:10:42)
mycommand (17:10:45)
mycommand (17:10:48)
mycommand (17:10:51)
mycommand (17:10:54)
mycommand (17:10:57)
```

Stop the container:

```
root@debian11:~/myDocker# docker stop -t 1 myDocker
myDocker
```

```
root@debian11:~/myDocker# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
NAMES
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   55 minutes ago   Up 55 minutes   27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   8 hours ago      Up 8 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   8 hours ago      Up 8 hours      0.0.0.0:81->80/tcp,
:::81->80/tcp   quirky_moore
```

Start the container:

```
root@debian11:~/myDocker# docker start myDocker
myDocker

root@debian11:~/myDocker# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
NAMES
97fe360bb1d6   i2tch/mydocker       "/entrypoint.sh myco…"   6 minutes ago    Up 5 seconds
myDocker
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   56 minutes ago   Up 56 minutes   27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   8 hours ago      Up 8 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   8 hours ago      Up 8 hours      0.0.0.0:81->80/tcp,
:::81->80/tcp   quirky_moore
```

Pause the container:

```
root@debian11:~/myDocker# docker start myDocker
myDocker

root@debian11:~/myDocker# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
```

```
NAMES
97fe360bb1d6   i2tch/mydocker       "/entrypoint.sh myco…"   6 minutes ago    Up 5 seconds
myDocker
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   56 minutes ago   Up 56 minutes   27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   8 hours ago      Up 8 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   8 hours ago      Up 8 hours      0.0.0.0:81->80/tcp,
:::81->80/tcp   quirky_moore
```

Unpause the container:

```
root@debian11:~/myDocker# docker unpause myDocker
myDocker

root@debian11:~/myDocker# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS              PORTS
NAMES
97fe360bb1d6   i2tch/mydocker       "/entrypoint.sh myco…"   7 minutes ago    Up About a minute
myDocker
880733c6bdc3   i2tch/mongodb2       "docker-entrypoint.s…"   57 minutes ago   Up 57 minutes       27017/tcp
mongo2
885f75b6aa57   ittraining/mongodb   "bash"                   8 hours ago      Up 8 hours
mongo
04d910a3c93d   nginx                "/docker-entrypoint.…"   8 hours ago      Up 8 hours
0.0.0.0:81->80/tcp, :::81->80/tcp   quirky_moore
```

Now launch the container with a parameter:

```
root@debian11:~/myDocker# docker rm -fv myDocker
myDocker

root@debian11:~/myDocker# docker run -d --name myDocker i2tch/mydocker "Up and Running"
fd5ac836f674fe0bf7b5056e851cd15e4762a5e41b05e00d384bede5234e1f5f
```

```
root@debian11:~/myDocker# docker logs myDocker
Up and Running (17:14:23)
Up and Running (17:14:26)
Up and Running (17:14:29)
Up and Running (17:14:32)
Up and Running (17:14:35)
Up and Running (17:14:38)
root@debian11:~/myDocker#
```

Change the value of the **myVariable** environment variable:

```
root@debian11:~/myDocker# docker rm -fv myDocker
myDocker

root@debian11:~/myDocker# docker run -d --name myDocker --env myVariable=1 i2tch/mydocker
a9e02a8bb39df9d5c84fc1d58643bc38c228b0562731792e2356a801b50a9a14

root@debian11:~/myDocker# docker logs myDocker
mycommand (17:15:35)
mycommand (17:15:36)
mycommand (17:15:37)
mycommand (17:15:38)
mycommand (17:15:39)
mycommand (17:15:40)
mycommand (17:15:41)
root@debian11:~/myDocker#
```

2.2 - Best practices related to Cache

Non-Idempotent Operations

Create a **bestp** directory and the following Dockerfile:

```
root@debian11:~/myDocker# cd ..

root@debian11:~# mkdir bestp

root@debian11:~# cd bestp

root@debian11:~/bestp# vi Dockerfile

root@debian11:~/bestp# cat Dockerfile
FROM ubuntu:latest
RUN date +%N > /tmp/moment
ENTRYPOINT ["more"]
CMD ["/tmp/moment"]
```

The Dokerfile contains a non-idempotent operation.

> **Important**: An idempotent operation is one that consistently produces the same result when run in the same context.

Compile the image:

```
root@debian11:~/bestp# docker build -t testcache .
[+] Building 0.9s (6/6) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.2s
 => => transferring dockerfile: 123B
0.0s
 => [internal] load .dockerignore
0.1s
 => => transferring context: 2B
```

```
0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest
0.0s
 => [1/2] FROM docker.io/library/ubuntu:latest
0.1s
 => [2/2] RUN date +%N > /tmp/moment
0.4s
 => exporting to image
0.1s
 => => exporting layers
0.1s
 => => writing image sha256:842ab4a40890a1b5fe7a3af5a41513c6edd5fd2da503b82c375f350671b62707
0.0s
 => => naming to docker.io/library/testcache
0.0s
```

Now run a first container from the compiled image:

```
root@debian11:~/bestp# docker run --name test1 -it testcache
771723987
```

Now remove the container and compile the image again:

```
root@debian11:~/bestp# docker rm test1
test1

root@debian11:~/bestp# docker build -t testcache .
[+] Building 0.3s (6/6) FINISHED
docker:default
 => [internal] load .dockerignore
0.1s
 => => transferring context: 2B
0.0s
 => [internal] load build definition from Dockerfile
```

```
0.1s
 => => transferring dockerfile: 123B
0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest
0.0s
 => [1/2] FROM docker.io/library/ubuntu:latest
0.0s
 => CACHED [2/2] RUN date +%N > /tmp/moment
0.0s
 => exporting to image
0.0s
 => => exporting layers
0.0s
 => => writing image sha256:842ab4a40890a1b5fe7a3af5a41513c6edd5fd2da503b82c375f350671b62707
0.0s
 => => naming to docker.io/library/testcache
0.0s
```

Launch a container from the re-compiled image:

```
root@debian11:~/bestp# docker run --name test1 -it testcache
771723987
```

Important - Note that the two container outputs are identical despite the fact that the value of the date command should have changed the result obtained when the second container was run. The reason this is not the case is the use of the cache in the second compilation. If this command had been something more important such as apt-get upgrade, the result could be embarrassing!

To get around this problem, it is possible to use the **–no-cache** option. Unfortunately this would produce a full compilation every time, even for

idempotent operations. It is therefore advisable to combine non-idempotent operations with idempotent operations in the same command line in order to invalidate the cache for that command line only:

```
root@debian11:~/bestp# vi Dockerfile

root@debian11:~/bestp# cat Dockerfile
FROM ubuntu:latest
RUN date +%N > /tmp/moment \
    && echo "V1.1" > /tmp/version
ENTRYPOINT ["more"]
CMD ["/tmp/moment"]
```

Now remove the container and start compiling the image again:

```
root@debian11:~/bestp# docker rm test1
test1

root@debian11:~/bestp# docker build -t testcache .
[+] Building 0.7s (6/6) FINISHED
docker:default
 => [internal] load .dockerignore
0.1s
 => => transferring context: 2B
0.0s
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 159B
0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest
0.0s
 => CACHED [1/2] FROM docker.io/library/ubuntu:latest
0.0s
 => [2/2] RUN date +%N > /tmp/moment     && echo "V1.1" > /tmp/version
0.4s
```

```
 => exporting to image
0.1s
 => => exporting layers
0.1s
 => => writing image sha256:5a36b1c7ec76e7bde962c41f5f5dcc11ae0ce3968e4953fbababcc8b7b282dab
0.0s
 => => naming to docker.io/library/testcache
0.0s
```

Launch a container from the re-compiled image:

```
root@debian11:~/bestp# docker run --name test1 -it testcache
063819144
```

LAB #3 - Installing a Private Registry

3.1 - Installing a Local Registry

To install a private registry, a public docker image should be used:

```
root@debian11:~/bestp# cd ..

root@debian11:~# docker run -d --name registry -p 88:5000 registry:latest
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
c926b61bad3b: Pull complete
5501dced60f8: Pull complete
e875fe5e6b9c: Pull complete
21f4bf2f86f9: Pull complete
98513cca25bb: Pull complete
Digest: sha256:0a182cb82c93939407967d6d71d6caf11dcef0e5689c6afe2d60518e3b34ab86
```

```
Status: Downloaded newer image for registry:latest
272df4a849bcbc58a70d6c8e1e74751f24e485fd8ad6817427ef180b9f28b5f8
```

Now use **lynx** from a terminal on your **Docker host** machine to check that the registry is active:

```
root@debian11:~# lynx --dump http://localhost:88/v2
{}root@debian11:~#
```

Important - Note the server response is **{}** either an empty JSON list.

Rename the **i2tch/mydocker** image to point to the new registry:

```
root@debian11:~# docker tag i2tch/mydocker localhost:88/mydocker
```

Send your **localhost:88/mydocker** image to this new registry:

```
root@debian11:~# docker push localhost:88/mydocker
Using default tag: latest
The push refers to repository [localhost:88/mydocker]
f981bd64e799: Pushed
74ddd0ec08fa: Pushed
latest: digest: sha256:32f7a11d8a8523bb5b4ac0986844d569ca96df4d1875e7e678a885ee3a3c61c3 size: 736
```

Now note the presence of the image in the registry:

```
root@debian11:~# lynx --dump http://localhost:88/v2/mydocker/tags/list
{"name":"mydocker","tags":["latest"]}
```

3.2 - Creating a Dedicated Registry Server

Currently, the private registry created above cannot be accessed from the local network because it is referenced by localhost. We now need to set up a dedicated server.

Connect to the **CentOS_10.0.3.45_SSH** VM from your **Debian_10.0.3.46_SSH** VM:

```
root@debian11:~# ssh -l trainee 10.0.3.45
trainee@10.0.3.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Nov 15 05:24:16 2023 from 10.0.3.1
[trainee@centos8 ~]$
```

Become root:

```
[trainee@centos8 ~]$ su -
Password: fenestros
[root@centos8 ~]#
```

Change the hostname of the machine:

```
[root@centos8 ~]# nmcli general hostname myregistry.i2tch.loc
[root@centos8 ~]# hostname
myregistry.i2tch.loc
```

Edit the **/etc/hosts** file and change the entry to the IP address 10.0.3.61:

```
[root@centos8 ~]# vi /etc/hosts
[root@centos8 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.3.45       myregistry.i2tch.loc
```

```
10.0.3.46       debian11.i2tch.loc
```

Now create a self-signed certificate with **openssl**:

```
[root@centos8 ~]# cd /

[root@centos8 /]# vi myconfig.cnf

[root@centos8 /]# cat myconfig.cnf
[ req ]
distinguished_name = dn
x509_extensions = extensions
prompt = no

[ extensions ]
subjectAltName = DNS:i2tch.loc,DNS:myregistry.i2tch.loc

[ dn ]
0.DC = loc
1.DC = i2tch
commonName = i2tch.loc

[root@centos8 ~]# mkdir certs && openssl req -config myconfig.cnf -newkey rsa:4096 -nodes -sha256 -keyout
certs/domain.key -x509 -days 365 -out certs/domain.crt
Generating a RSA private key
.....................................................................................................
.....................................................................+++
......+++
writing new private key to 'certs/domain.key'
-----

[root@centos8 /]# ls certs/
domain.crt domain.key
```

Disconnect from the **CentOS8_10.0.3.45_SSH** VM:

```
[root@centos8 /]# exit
logout
[trainee@centos8 ~]$ exit
logout
Connection to 10.0.3.45 closed.
root@debian11:~#
```

Reconnect to the **CentOS8_10.0.3.45_SSH** VM:

```
root@debian11:~# ssh -l trainee 10.0.3.45
trainee@10.0.3.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Dec 15 01:07:37 2023 from 10.0.3.46
[trainee@centos8 ~]$
```

Become root:

```
[trainee@myregistry ~]$ su -
Password: fenestros
[root@myregistry ~]#
```

Create a container in secure mode with TLS from the registry image:

```
[root@myregistry ~]# docker run -d -p 5000:5000 --name registry -v /certs:/certs -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key registry:latest
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
c926b61bad3b: Pull complete
5501dced60f8: Pull complete
e875fe5e6b9c: Pull complete
21f4bf2f86f9: Pull complete
```

```
98513cca25bb: Pull complete
Digest: sha256:0a182cb82c93939407967d6d71d6caf11dcef0e5689c6afe2d60518e3b34ab86
Status: Downloaded newer image for registry:latest
bf0d4fe9fcb121f9c2d9e85b8f2bb54b01397602ef0dcefdfc71327acf832fec

[root@myregistry ~]# docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                      PORTS
NAMES
bf0d4fe9fcb1   registry:latest   "/entrypoint.sh /etc…"   47 seconds ago   Up 44 seconds
0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   registry
90267aac9800   hello-world       "/hello"                 15 hours ago     Exited (0) 15 hours ago
eloquent_chatelet
```

Send a copy of the **/certs/domain.crt** file to the /tmp directory of the **Debian11_10.0.3.46** virtual machine, renaming it **ca.crt**:

```
[root@myregistry ~]# scp /certs/domain.crt trainee@10.0.3.46:/tmp/ca.crt
The authenticity of host '10.0.3.46 (10.0.3.46)' can't be established.
ECDSA key fingerprint is SHA256:JFem/0UXFw0aDAOSfOS3vsOGsSDl1wPOza6ybTGO7/8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.3.46' (ECDSA) to the list of known hosts.
trainee@10.0.3.46's password: trainee
domain.crt 100% 2053 2.9MB/s 00:00
```

Configuring the Client

Exit the VM **CentOS8_10.0.3.45_SSH**:

```
[root@myregistry ~]# exit
logout
[trainee@myregistry ~]$ exit
logout
Connection to 10.0.3.45 closed.
```

```
root@debian11:~#
```

Remove the **registry** container:

```
root@debian11:~# docker rm -f registry
registry
```

As well as the registry image:

```
root@debian11:~# docker rmi registry:latest
Untagged: registry:latest
Untagged: registry@sha256:0a182cb82c93939407967d6d71d6caf11dcef0e5689c6afe2d60518e3b34ab86
Deleted: sha256:909c3ff012b7f9fc4b802b73f250ad45e4ffa385299b71fdd6813f70a6711792
Deleted: sha256:577c3b283118ca6108a6a8c8a0a00eff666dec82c482dd239dfed49f31553df6
Deleted: sha256:2ba6acf6ed95c86cfb2c830693135513bc019a0c0cf8f2c58990bc215995699f
Deleted: sha256:65920463e77382a5cbe8da3e814c4449fc665487c8a9fa4ac27179e809f5ba2e
Deleted: sha256:54501ccbeaec2665849d200fc4a61ab7254ff0f3bd31ab673879fe321fa2ad7f
Deleted: sha256:9fe9a137fd002363ac64f5af66146702432b638a83ee0c5b620c40a9e433e813
```

Rename the **i2tch/mydocker** image to point to the registry server:

```
root@debian11:~# docker tag i2tch/mydocker myregistry.i2tch.loc:5000/mydocker

root@debian11:~# docker images
REPOSITORY                           TAG       IMAGE ID       CREATED        SIZE
testcache                            latest    5a36b1c7ec76   4 days ago     77.8MB
<none>                               <none>    842ab4a40890   4 days ago     77.8MB
i2tch/mydocker                       latest    c5a41438d278   4 days ago     231MB
localhost:88/mydocker                latest    c5a41438d278   4 days ago     231MB
myregistry.i2tch.loc:5000/mydocker   latest    c5a41438d278   4 days ago     231MB
i2tch/mongodb2                       latest    56e5b1fb4284   4 days ago     352MB
i2tch/mongodb1                       latest    72fad0b7e0c2   4 days ago     352MB
ittraining/mongodb                   latest    fb3c6d5d186a   5 days ago     1.11GB
ubuntu                               latest    b6548eacb063   2 weeks ago    77.8MB
```

```
nginx                                 latest    a6bd71f48f68   3 weeks ago    187MB
hello-world                           latest    9c7a54a9a43c   7 months ago   13.3kB
centos                                latest    5d0da3dc9764   2 years ago    231MB
```

Edit the **/etc/hosts** file to point the 10.0.3.45 to the name **myregistry.i2tch.loc**:

```
root@debian11:~# vi /etc/hosts

root@debian11:~# cat /etc/hosts
127.0.0.1       localhost
10.0.3.46       debian11.i2tch.loc      debian11
10.0.3.45       myregistry.i2tch.loc    myregistry

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Move the **/tmp/ca.crt** file to the **/etc/docker/certs.d/myregistry:5000/** directory:

```
root@debian11:~# mkdir -p /etc/docker/certs.d/myregistry:5000

root@debian11:~# mv /tmp/ca.crt /etc/docker/certs.d/myregistry:5000/
```

Create the **/etc/docker/daemon.json** file to accept the self-signed certificate:

```
root@debian11:~# vi /etc/docker/daemon.json

root@debian11:~# cat /etc/docker/daemon.json
{"insecure-registries" : ["myregistry.i2tch.loc:5000"]}
```

Restart the docker service:

```
root@debian11:~# systemctl restart docker
```

Test the registry response:

```
root@debian11:~# curl -k https://myregistry:5000/v2/
{}root@debian11:~#
```

Finally, send the image to the registry:

```
root@debian11:~# docker push myregistry.i2tch.loc:5000/mydocker
Using default tag: latest
The push refers to repository [myregistry.i2tch.loc:5000/mydocker]
f981bd64e799: Pushed
74ddd0ec08fa: Pushed
latest: digest: sha256:32f7a11d8a8523bb5b4ac0986844d569ca96df4d1875e7e678a885ee3a3c61c3 size: 736
```
